VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

ELEKTRONICKÝ REZERVAČNÍ SYSTÉM PRO LABORATOŘE A LABORATORNÍ ZAŘÍZENÍ

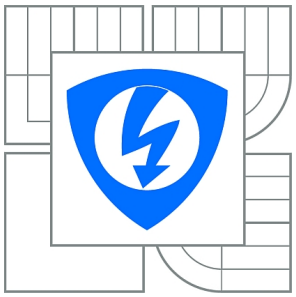
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR
LUKÁŠ POVODA

BRNO 2012

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF TELECOMMUNICATIONS

ELEKTRONICKÝ REZERVAČNÍ SYSTÉM PRO LABORATOŘE A LABORATORNÍ ZAŘÍZENÍ

ELECTRONIC BOOKING SYSTEM FOR LABORATORIES AND LABORATORY EQUIPMENT

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

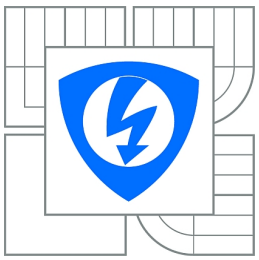
AUTOR PRÁCE — LUKÁŠ POVODA

AUTHOR

VEDOUCÍ PRÁCE — Ing. JIŘÍ SCHIMMEL, Ph.D.

SUPERVISOR

BRNO 2012



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Bakalářská práce

bakalářský studijní obor
Teleinformatika

Student:	Lukáš Povoda	***ID:***	125607
Ročník:	3	***Akademický rok:***	2011/2012

NÁZEV TÉMATU:

Elektronický rezervační systém pro laboratoře a laboratorní zařízení

POKYNY PRO VYPRACOVÁNÍ:

Navrhněte a realizujte databázový systém s webovým rozhraním pro rezervování laboratoří a laboratorních zařízení. Systém bude přehlednou grafickou a interaktivní formou umožňovat administrátorům přidávání položek a správu účtů, registrovaným uživatelům rezervaci laboratoří a zařízení a zobrazení zápůjčních listů a všem uživatelům zobrazení a tisk přehledů obsazení laboratoří ve formě detailního plánovacího kalendáře. Systém bude využívat databáze MySQL a jazyka PHP.

DOPORUČENÁ LITERATURA:

[1] J. Castagnetto, H. Rawat, S. Schumann, C. Scollo, D. Veliath, Programujeme PHP profesionálně. Computer Press, Praha, 2001. ISBN 80-7226-310-2
[2] V. Pošmura, Apache příručka správce www serveru. Computer Press, Praha, 2002. ISBN 80-7226-696-9
[3] M. Šimůnek, SQL kompletní kapesní průvodce. Grada Publishing spol. s r.o., 1999. ISBN 80-7169-692-7

Termín zadání:	6.2.2012	***Termín odevzdání:***	31.5.2012

Vedoucí práce: Ing. Jiří Schimmel, Ph.D.
Konzultanti bakalářské práce:

prof. Ing. Kamil Vrba, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

ABSTRAKT

Práca sa zaoberá návrhom databázovej štruktúry a webového rozhrania rezervačného systému, určeného pre rezerváciu laboratórnych priestorov a meracích zariadení určených pre zapožičanie. Systém je navrhnutý na najrozšírenejšom interpretovanom jazyku určenom pre tvorbu webových aplikácií – na jazyku PHP, a na databázovom systéme MySQL, konkrétne na úložnom systéme InnoDB určenom pre relačné databázy a transakcie. V práci je vysvetlený princíp činnosti moderných návrhových vzorov Model–View–Presenter a Dependency injection, a ich praktické využitie. Samotný kód webovej aplikácie je postavený na PHP frameworku Nette, PHP knižnici dibi určenej aj pre databázy MySQL, knižnici mPDF určenej pre generovanie PDF súborov z HTML kódu. Celý systém je navrhnutý podľa konvencií frameworku Nette a spomínaných návrhových vzorov, takže ďalšie rozširovanie, úpravy implementovaných funkcií, úpravy dizajnu webového rozhrania, práca s dátami, či implementácia nových rozhraní a funkcií bude jednoduchá a časovo menej náročnejšia, ako pri systémoch bez zaužívaných konvencií a návrhových vzoroch.

KĽÚČOVÉ SLOVÁ

Rezervačný systém, databáza, návrhové vzory, kalendár, zapožičiavací list

ABSTRACT

The main aim of this paper is to create the database structure and the web interface of the reservation system. This system is for the reservation of the laboratories and of the measuring equipement (for hire). System was created by the most widely language intended for creating Web application – language PHP, and database system MySQL, specifficaly base on the storage system InnoDB formed for relation database and the transaction. This paper descbires the princip of the work of modern patterns like Model–View–Presenter or Dependency injection and their both practical use. The code of the web application has been based on PHP framework Nette, on PHP library dibi created for database MySQL, on library mPDF created for generation of the PDF files from HTML code. System has been created according to convention of framework Nette and already mentioned design pattern, so it means that all changes such as spreading, changes of implemented funcions, changes of design of web interface, work with the datas or implemenation of the new interface and of new funcions will be simpler and take less time then for example the use of the system without convention and pattern.

KEYWORDS

Reservation system, database, patterns, calendar, device-lending list

POVODA, Lukáš *Elektronický rezervačný systém pre laboratóriá*: bakalárska práca. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2011. 47 s. Vedúci práce bol Ing. Jiří Schimmel, Ph.D.

PREHLÁSENIE

Prehlasujem, že som svoju bakalársku prácu na tému „Elektronický rezervačný systém pre laboratóriá“ vypracoval samostatne pod vedením vedúceho bakalárskej práce, využitím odbornej literatúry a ďalších informačných zdrojov, ktoré sú všetky citované v práci a uvedené v zozname literatúry na konci práce.

Ako autor uvedenej bakalárskej práce ďalej prehlasujem, že v súvislosti s vytvorením tejto bakalárskej práce som neporušil autorské práva tretích osôb, najmä som nezasiahol nedovoleným spôsobom do cudzích autorských práv osobnostných a/nebo majetkových a som si plne vedomý následkov porušenia ustanovenia § 11 a nasledujúcich autorského zákona č. 121/2000 Sb., o právu autorském, o právoch súvisejúcich s právom autorským a o zmeně niektorých zákonov (autorský zákon), vo znení neskorších predpisov, vrátane možných trestnoprávnych dôsledkov vyplývajúcich z ustanovenia časti druhé, hlavy VI. diel 4 Trestného zákoníka č. 40/2009 Sb.

Brno ....

(podpis autora)

OBSAH

ZOZNAM OBRÁZKOV

ZOZNAM TABULIEK

ÚVOD

Táto práca je založená na konkrétnych požiadavkách pre vytvorenie databázového systému s kompletným webovým rozhraním, určeného pre rezervovanie laboratórií a meracích prístrojov. Systém je navrhnutý pre úložný engine InnoDB obsiahnutý v databázových systémoch MySQL. O logiku samotného webového rozhrania sa bude starať interpretovaný skriptovací jazyk PHP.

V tejto práci by som chcel demonštrovať využitie viacvrstvového návrhového vzoru Model–View–Presenter (MVP), návrhového vzoru určeného pre logiku webových aplikácií – Dependency Injection (DI), použitie moderných šablónovacích systémov, a ich kompletnú funkčnosť nad databázami MySQL.

Pre tvorbu tejto aplikácie som použil framework Nette – open source framework vyvíjaný českými a slovenskými vývojármi zaoberajúcimi sa jazykom PHP. Nette je postavené na spomínaných návrhových vzoroch a eliminuje bezpečnostné riziká, ktoré môžu nastať pri webových aplikáciách, ako napr. XSS, CSRF, session hijacking, session fixation a pod., ako píše zakladateľ frameworku na oficiálnych stránkach. [1]

1 DATABÁZA S WEBOVÝM ROZHRANÍM

1.1 Návrh databáz

Každá databáza sa skladá z jednej alebo viac tabuliek. Jeden riadok tejto tabuľky nám predstavuje jeden daný záznam, jeden stĺpec nám predstavuje množinu údajov rovnakého dátového typu, teda určitú vlastnosť daných záznamov. [2]

1.1.1 Relačné databázy

Štruktúra typu riadok – stĺpec nám však málokedy stačí a tak potrebujeme vytvoriť aj závislosti medzi jednotlivými tabuľkami, napr. užívateľ – oprávnenie – sekcia prístupu. Takúto závislosť vytvoríme ľahko a jednoducho v relačných databázach pomocou cudzích kľúčov. Cudzie kľúče nám pomáhajú pri návrhu databáz a zrýchľujú prácu s dátami, hlavne pri databázach s obrovskými počtami záznamov. Zároveň zabraňujú chybnému vloženiu údajov, tj. cudzích kľúčov ktoré neexistujú. V MySQL nám túto funkcionalitu relačných databáz zabezpečuje úložný systém InnoDB. Na rozdiel od iných úložných systémov umožňuje využívanie transakcií, teda vykonanie určitého zhluku príkazov iba ak v žiadnom z nich nenastane chyba. Ak chyba nastáva, zmeny sa v databáze nezapíšu. [3] [4]

1.1.2 Normálové formy

Pri návrhu štruktúry samotnej databázy sa odporúča prihliadať určitým súborom doporučení. Tieto sa nazývajú normálové formy. Správne navrhnuté databázové tabuľky spĺňujú štyri základné normálové formy:

Prvá normálová forma je splnená, ak sú všetky vlastnosti záznamu atomické, teda sú nedeliteľné. Dobrým príkladom môže byť napríklad adresa bydliska užívateľa. Pokiaľ chceme dodržať 1NF, musíme údaje o bydlisku uložiť ako samostatné stĺpce tabuľky – teda ulica, číslo domu, mesto a poštové smerovacie číslo.

Druhá normálová forma je splnená, ak každá vlastnosť, okrem primárneho kľúča, je na primárnom kľúči úplne závislá. Teda ak máme tabuľku so zoznamom rezervovaných miestností, môže sa vyskytnúť prípad, že užívateľ bude mať registrovanú istú miestnosť viac krát do týždňa.

Tab. 1.1: Tabuľka rezervácií pred aplikovaním 2NF

Login	Čas od	Čas do	Číslo dverí	Miestnosť
xlogin00	14.11.2011 8:00	14.11.2011 12:00	PA-322A	Bezodrazová komora
xuziva00	14.11.2011 12:00	14.11.2011 16:00	PA-136	Akustická laboratoř
xlogin00	15.11.2011 14:00	15.11.2011 18:00	PA-322A	Bezodrazová komora

Ak chceme však dodržať 2NF, musíme údaje rozdeliť do dvoch tabuliek. A síce do tabuľky rezervácie a do tabuľky miestnosti. Následne ich potom prepojíme pomocou cudzieho kľúča z tabuľky miestnosti. Údaje potom budú uložené v tabuľkách napríklad takto[1]:

Tab. 1.2: Príklad aplikácie 2NF - tabuľka rezervácie

Login	Čas od	Čas do	ID miestnosti
xlogin00	14.11.2011 8:00	14.11.2011 12:00	2
xuziva00	14.11.2011 12:00	14.11.2011 16:00	1
xlogin00	15.11.2011 14:00	15.11.2011 18:00	2

Tab. 1.3: Príklad aplikácie 2NF - tabuľka miestnosti

ID	Číslo dverí	Miestnosť
1	PA-136	Akustická laboratoř
2	PA-322A	Bezodrazová komora

Tretia normálová forma je splnená, ak každá vlastnosť, okrem primárneho kľúča, nie je tranzitívne závislá na primárnom kľúči. Napríklad ak si užívateľ zapožičia nejaký merací prístroj, dostane ho i s príslušenstvom. Máme teda závislosť užívateľ → prístroj → príslušenstvo. Aby sme dodržali 3NF, musíme toto rozdeliť do troch tabuliek[2].

Boyce-Coddová normálová forma je splnená, ak pre dve množiny vlastností **A** a **B** platí $\mathbf{A} \in \mathbf{B}$ a súčasne $\mathbf{B} \notin \mathbf{A}$, potom množina **A** obsahuje primárny kľúč. Inými slovami, ak pri návrhu postupne splňujeme všetky normálové formy, bude splnená i BCNF.

Všetky normálové formy a konkrétne príklady sú uvedené v článku J. Skřivana [5] a v článku J. Kulhana [6].

[1] Jedná sa o príklad použitia 2NF, skutočné riešenie danej úlohy je popísané v kapitole 2.2

[2] Ide len o teoretický príklad. Príslušenstvo v tabuľke prístrojov nie je atomický údaj, teda nesplňuje ani 1NF.

1.2 Webové rozhranie

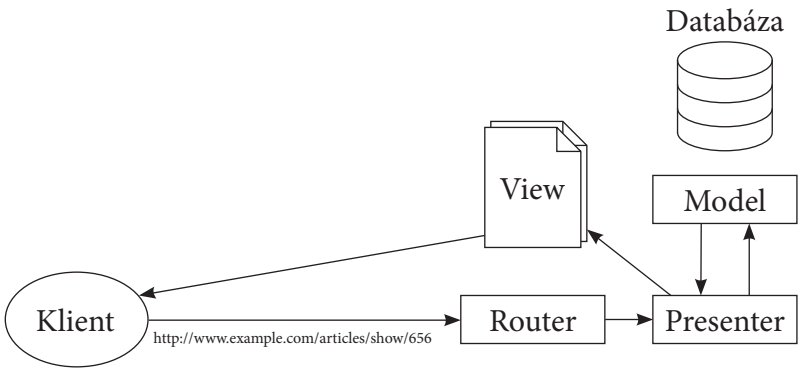
Pre vývoj webových aplikácií sa najviac používa jazyk PHP. Už od verzie 4 začali vznikať knižnice so zovšeobecnenými funkciami a triedami. Populárne sa stali aj šablónovacie systémy pre jednoduchšie sadenie značkovacieho jazyka HTML, napr. šablónovací systém Smarty, ktorý sa používa dodnes. Vo verzií PHP 5 pribudla lepšia podpora OOP a tak začali vznikať komplexnejšie nástroje pre vývoj webových aplikácií – označované ako frameworky. [7]

1.2.1 PHP frameworky

„Framework je softwarová štruktúra, ktorá slúži ako podpora pri vývoji a organizácií iných softwarových projektov.“ [8] Ich cieľom je zjednodušenie práce, a to tak, že daný problém uvažujeme na vyšších abstraktných úrovniach. Väčšina PHP frameworkov k tomu využíva návrhové vzory MVC, respektíve MVP ako je tomu u frameworku Nette.

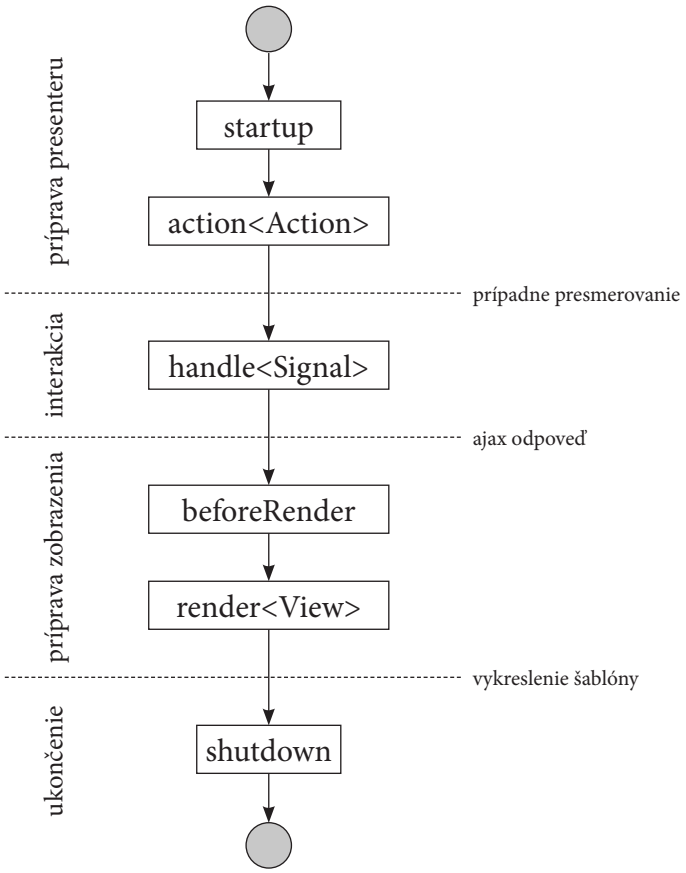
1.2.2 Návrhový vzor MVP

Model–View–Presenter je softwarová architektúra, ktorá rozdeľuje webovú aplikáciu na viac abstraktných úrovní – na úroveň obsluhy (presenter), úroveň prístupu k dátam (model), a úroveň zobrazujúcu dáta (view). Pomocou tohto vzoru sa kód aplikácie stáva udržiavateľným, úpravy sú omnoho jednoduchšie a rýchlejšie, vývoj je extrémne rýchly. Oddelenie logiky a vzhľadu tak umožňuje vývojárom a dizajnérom pracovať súčasne. [9] [10]



Obr. 1.1: Diagram návrhového vzoru Model–View–Presenter.

Na obr. 1.1 je graficky znázornený proces spracovávania požiadavky pre zobrazenie stránky. Klient si vyžiada určitý obsah podľa URL adresy, z ktorej potom router (niekedy nazývaný aj dispečer) zistí, ktorý presenter sa použije, a následne ktorá z jeho akcií bude zavolaná. Presenter prostredníctvom modelov môže pristupovať k dátam uloženým v jednej alebo viacerých databázach. Tieto dáta následne spracuje a predá na ďalšiu úrovneň – view. Na tejto úrovni sa dáta dosadia do šablóny, kde na výstupe dostaneme buď hotový HTML kód, XML výstup, JSON apod. V šablóne môžeme využiť helpery a rôzne makra, ktoré nám formátovanie výsledného kódu uľahčia. Pokiaľ však dôjde napr. k neoprávnenému prístupu, môže presenter predčasne ukončiť svoju činnosť a presmerovať klienta na inú URL adresu. Toto možno využiť už v metóde startup, kde takto môžeme ľahko kontrolovať oprávnenia užívateľov. V takomto prípade je najlepšie vzťahovať oprávnenia priamo na názvy presenterov.



Obr. 1.2: Životný cyklus presenteru.

Pokiaľ máme za úlohu vytvoriť určitú dynamickú časť stránky, napr. našepkávanie a vypĺňanie prvkov formulára dátami uloženými v databáze (použité v automatickom vypĺňaní adresy budovy), môžeme pre to využiť spracovanie signálov, ktoré

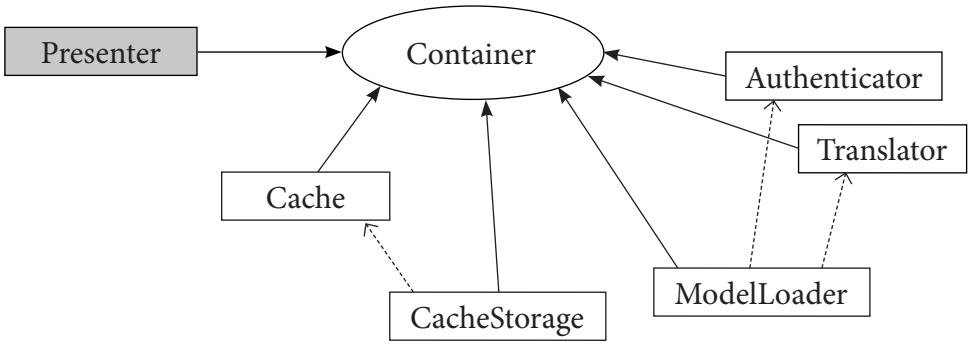
sa prevádza v metóde `handle<NázovSignálu>`. Celý životný cyklus presenteru je znázornený na obr. 1.2, so všetkými metódami volanými pri spracovaní, plus možné využitia jednotlivých krokov. Prebrané z [9] – podrobnejšie informácie a oficiálna dokumentácia.

1.2.3 Návrhový vzor DI

Tento návrhový vzor sa pôvodne presadil hlavne v jazyku Java, no s rozšírenými možnosťami OOP[3] si našiel svojich priaznivcov aj v jazyku PHP. Dependency Injection slúži ako vzor pre zníženie závislosti medzi jednotlivými časťami systému, tak aby jeden komponent mohol používať iný, bez toho aby v dobe zostavovania jedného komponentu existovala referencia na druhý komponent. [11][12]

1.2.4 Služby vo frameworku Nette

Ako príklad využitia DI môžeme zobrať služby (services) vo frameworku Nette. Služby sú isté globálne objekty, ktoré možno používať v celom frameworku. Tieto objekty vznikajú a inicializujú sa až keď je daná služba naozaj požadovaná. Z funkcionality DI vyplýva, že ak jedna služba požaduje službu inú, vytvorí sa najskôr služba požadovaná. Takto vytvorené sa uložia do kontajneru zvaného `context`. Pomocou neho môžu presentery pristupovať ku všetkým službám. [13]



Obr. 1.3: Príklad použitia DI.

Ako príklad môžeme vziať službu `Authenticator` z obr. 1.3. Táto služba slúži pre autentifikáciu užívateľa, testuje správnosť zadaných prihlasovacích údajov a obsahuje metódu pre zašifrovanie hesla s použitím soli.[4] Už pri vzniku tohto objektu

[3] Od PHP verzie 5

[4] Takto presne funguje aj autentifikátor vo finálnom riešení.

sa však požaduje, aby existovala služba `ModelLoader`, pomocou ktorého pristupuje k údajom v databáze. Toto si framework zariadi sám vďaka spomínanému DI.

Služby, ich triedy a ich vstupné parametre sa definujú v súbore `config.neon`, ktorý má presne danú štruktúru a zápis, odvodený od jazyka YAML[5].

```
common:
    # ...
    services:
        robotLoader:
            run: true
        model:
            class: Models\ModelsLoader
            arguments: [%database%]
        authenticator:
            class: Models\Authenticator
            arguments: [@model, user]
    # ...
```

Obr. 1.4: Príklad konfigurácie služieb.

Pomocou takto nakonfigurovaného prostredia sa nám v kontexte vytvoria služby `model` a `autheticator`, ktorých vstupné parametre označené znakom percento sú brané ako premenné definované v súbore `config.neon`, a parametre označené znakom zavináč sú brané ako referencie na iné služby. Na obr. 1.4 je aj služba `robotLoader`, ktorá nevyhnutná pre správnu funkciu frameworku – zaisťuje načítavanie knižníc a tried presenterov, modelov apod.

1.2.5 Šablónovací systém Latte

Jazyk PHP bol pôvodne navrhnutý ako šablónovací jazyk, no k ich kódovaniu nie je vhodný. Preto sa časom začali vytvárať šablónovacie systémy ako XSL, Smarty apod. Časom sa vypracovali lepšie, prehľadnejšie a omnoho použiteľnejšie šablónovacie systémy. Jedným z nich je práve latte.

```
<ul n:if="$items">
{foreach $items as $item}
    <li id="item-{$iterator->counter}">{$item|capitalize}</li>
{/foreach}
</ul>
```

Obr. 1.5: Príklad zápisu latte šablóny.

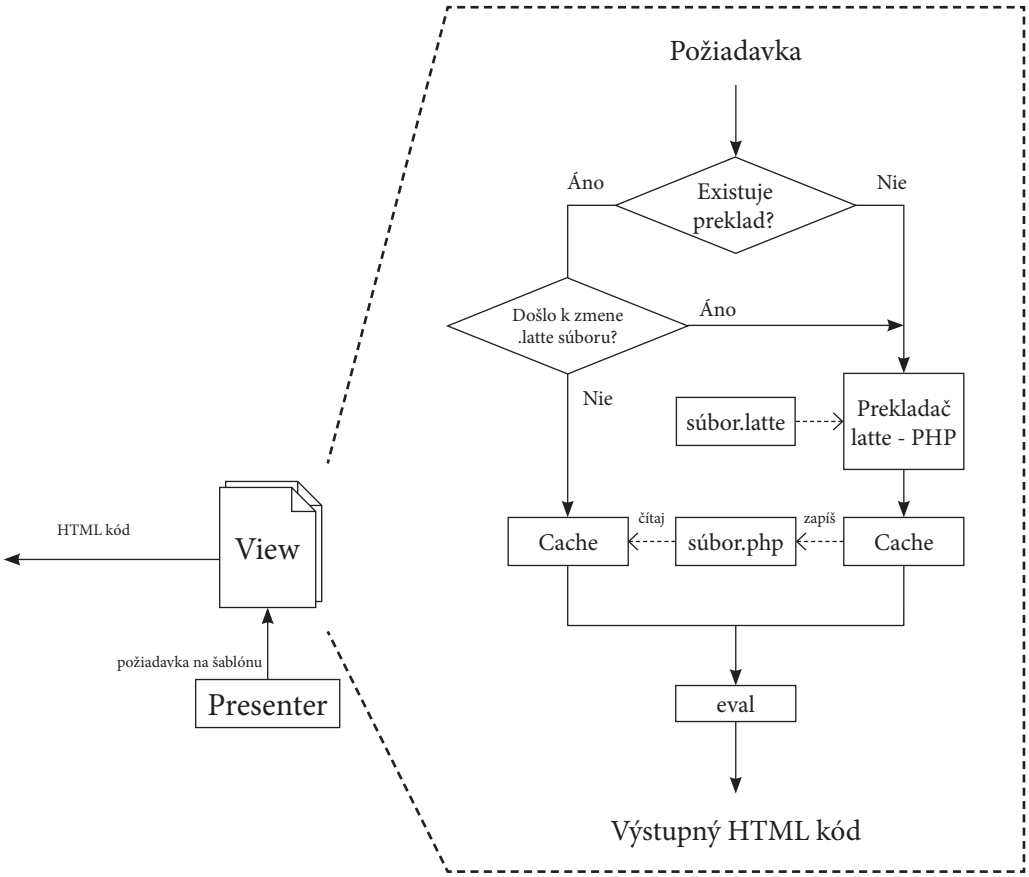
V latte šablónach možno využívať makra `if`, `for`, `foreach` a iné. Takisto sa jednoducho dajú definovať ďalšie, ktorými si opäť uľahčíme prácu a sprehľadníme kód.

[5]Značkovací jazyk ľahko čitateľný pre človeka.

Makra majú aj podobu n:makra, kedy sa stávajú súčasťou HTML atribútov jednotlivých elementov. V šablónach taktiež môžme volať helpery, ktoré nám preberajú funkciu metód, a tak môžeme premennú ešte poslať na spracovanie do PHP kódu, pričom výsledok je opäť HTML kód. Okrem helperov môžeme používať rôzne PHP objekty, zvané komponenty. Tieto sa starajú napr. o vykreslenie formulára.

Takáto šablóna vôbec nespomaľuje spracovanie. Pri zavolaní je preložená do PHP kódu len raz (respektíve pri každej zmene šablóny), ktorý je následne uložený do pamäte cache. Ak sa opäť požaduje použitie šablóny, je využívaný už preložený kód v jazyku PHP, ktorý ide priamo na spracovanie. [14]

Dáta na zobrazenie sú posielané do šablóny z presenteru prostredníctvom premennej template. Celý postup spracovania latte šablóny je na obr. 1.6.



Obr. 1.6: Spracovanie latte šablóny.

2 TVORBA REZERVAČNÉHO SYSTÉMU

2.1 Rozbor zadania

Mojou úlohou bolo navrhnúť databázový systém s webovým rozhraním pre rezervovanie laboratórií a meracích zariadení. V systéme je možné laboratóriá a meracie prístroje pridávať, zobrazovať ich zoznam, filtrovať ich podľa určitých kritérií, rezervovať a zobrazovať kalendár jednotlivých rezervácií. Systém prístupu k týmto akciám som rozdelil do štyroch úrovní[1], ktoré sú priradené konkrétnym prihláseným užívateľom – hostia, užívatelia, správcovia a administrátori. Neprihláseným užívateľom sa zobrazí iba prihlasovacia obrazovka[2] alebo možnosť registrácie. V systéme má byť tiež implementovaná jednoduchá administrácia pre správu registrovaných užívateľov a pre pridávanie záznamov – a síce pre pridávanie laboratórií a pridávanie meracích zariadení.

2.2 Návrh databázy

Databáza je postavená na systéme InnoDB, ktorý je súčasťou databázy MySQL. Vďaka tomuto systéme sa dajú tabuľky prepojiť cudzími kľúčmi a vytvoriť tak relácie medzi jednotlivými tabuľkami. Zároveň máme zaistené, že vložené cudzie kľúče naozaj existujú ako primárny kľúč v inej tabuľke.

2.2.1 Tabuľky

V tab. 2.1 sa nachádza kompletný zoznam vytvorených databázových tabuliek, aj s popisom ich funkcií.

Tab. 2.1: Zoznam databázových tabuliek

Názov tabuľky	**Účel tabuľky**
building	Adresy budov, v ktorých sa nachádzajú laboratóriá
device	Meracie zariadenia
device_reservation	Rezervácie meracích zariadení
room	Laboratóriá
room_reservation	Rezervácie laboratórií
user	Registrovaný užívatelia

[1] Nazvané ako užívateľské role, bližšie popísané v kapitole 2.3.2.

[2] Kvôli prípadnej integrácií do iných webových stránok.

Tab. 2.2: Štruktúra databázovej tabuľky s adresami budov

building

Názov stĺpca	Dátový typ	Popis
id	int(10) unsigned	Identifikačné číslo záznamu, primárny kľúč
street	int(10) unsigned	Názov ulice
number	varchar(30)	Číslo domu – popisné

Adresa budovy sa uloží do samostatnej tabuľky (popis viď tab. 2.2), aby sa znížil počet ukladaných údajov. Takto sa daná adresa uloží v databáze len raz. Následne sa do tabuľky s laboratóriami vloží len ID tejto adresy[3], viď tab. 2.3.

Tab. 2.3: Štruktúra databázovej tabuľky laboratórií

room

Názov stĺpca	Dátový typ	Popis
id	int(10) unsigned	Identifikačné číslo záznamu, primárny kľúč
building_id	int(10) unsigned	ID adresy budovy, kde sa laboratórium nachádza, index
door_number	varchar(30)	Číslo dverí, prípadne celé textové označenie dverí
name	varchar(255)	Celý názov laboratória
responsible_person	varchar(255)	Zodpovedná osoba, iba textom
manager_id	int(10) unsigned	ID užívateľa, ktorý záznam vytvoril, index

Ukladanie jednotlivých rezervácií sa deje do samostatnej tabuľky (štruktúra viď tab. 2.4). Takto možno ľahko realizovať výbery ako napr. zoznam rezervácií jedného užívateľa, všetky rezervácie daného laboratória, rezervácie v daný deň apod.

Tab. 2.4: Štruktúra databázovej tabuľky rezervácií laboratórií

room_reservation

Názov stĺpca	Dátový typ	Popis
id	int(10) unsigned	Identifikačné číslo záznamu, primárny kľúč
room_id	int(10) unsigned	ID laboratória, ktoré si užívateľ rezervoval, index
user_id	int(10) unsigned	ID užívateľa, ktorý rezerváciu previedol, index
datetime_from	datetime	Dátum a čas, od kt. bude miestnosť obsadená
datetime_to	datetime	Dátum a čas, kedy užívateľ opustí miestnosť
purpose	tinytext	Dôvod rezervácie, vyplňuje užívateľ
comment	tinytext NULL	Komentár k rezervácií, vyplňuje užívateľ
status	enum(new,confirmed,refused)	Stav rezervácie, index

Do tabuľky musíme teda uložiť dátum a rozsah hodín, kedy bude laboratórium rezervované. Najjednoduchšie bude zvoliť dátový typ `datetime`, pomocou ktorého možno prevádzať operácie porovnávania, a tak môžeme dotazy pokladať ako by sme

[3]Toto prevádza databázový model Room, bližšie popísaného v kapitole 2.2.5.

pracovali napríklad s dvomi celočíselnými hodnotami. Zároveň správca databázy vidí ihneď hodnotu dátumu a času, a môže sa tam ľahko orientovať v jednotlivých záznamoch. Stĺpec `status` nám predstavuje stav rezervácie, bližšie popísané v kap. 2.2.3.

Rovnako som navrhol aj tabuľku pre rezerváciu meracích zariadení (viď tab. 2.6), ktoré sú uložené v tabuľke meracích zariadení (viď tab. 2.5).

Tab. 2.5: Štruktúra databázovej tabuľky meracích zariadení

device

Názov stĺpca	Dátový typ	Popis
id	int(10) unsigned	Identifikačné číslo záznamu, primárny kľúč
name	varchar(255)	Názov meracieho zariadenia
specification	mediumtext NULL	Špecifikácia
accessories	mediumtext NULL	Príslušenstvo
evid_number	varchar(255)	Evidenčné číslo
evid_room	varchar(255)	Miestnosť, v ktorej je zariadenie evidované
evid_person	varchar(255)	Osoba, pod ktorou je zariadenie evidované
evid_name	varchar(255)	Názov majetku
evid_year	mediumint(4) unsigned NULL	Rok zakúpenia
evid_serial	varchar(255)	Sériové číslo
evid_comment	tinytext	Komentár k evidencií
price	bigint(20)	Cena zariadenia
manager_id	int(10) unsigned	ID užívateľa, ktorý záznam vytvoril, index

Tab. 2.6: Štruktúra databázovej tabuľky rezervácií meracích zariadení

device_reservation

Názov stĺpca	Dátový typ	Popis
id	int(10) unsigned	Identifikačné číslo záznamu, primárny kľúč
device_id	int(10) unsigned	ID zariadenia, ktoré si užívateľ rezervoval, index
user_id	int(10) unsigned	ID užívateľa, ktorý rezerváciu previedol, index
date_from	date	Dátum, od ktorého bude zariadenie obsadené
date_to	date	Dátum, vrátenia zariadenia zodpovednej osobe
purpose	tinytext	Dôvod rezervácie zariadenia, vyplňuje užívateľ
comment	tinytext NULL	Prípadný komentár k rezervácií, vyplňuje užívateľ
status	enum(new,confirmed,refused)	Stav rezervácie, index

Registrácie užívateľov prevádzajú samotní užívatelia pomocou registračného formulára. Z tohto dôvodu im pridelíme na začiatku užívateľskú rolu hosť. Administrátor potom môže na požiadanie rolu zmeniť. Pre označenie užívateľskej role využijeme stĺpček role v databázovej tabuľke užívateľov, ktorý bude typu enum. Jedná sa o typ s vymenovanými hodnotami, teda môže nadobúdať iba jednu hodnotu zo zoznamu vymenovaných – GUEST, USER, MANAGER alebo ADMIN. Celá štruktúra tabuľky registrovaných užívateľov sa nachádza v tab. 2.7.

Tab. 2.7: Štruktúra databázovej tabuľky s užívateľmi

user

Názov stĺpca	Dátový typ	Popis
id	int(10) unsigned	Číslo záznamu, primárny kľúč
login	varchar(20)	Login užívateľa
password	varchar(32)	Zašifrovaný hash hesla
role	enum(GUEST,USER,MANAGER,ADMIN)	Užívateľská rola
name	varchar(255)	Krstné meno
surname	varchar(255)	Priezvisko
email	varchar(255)	E-mailová adresa
phone	varchar(15)	Telefónne číslo
address	tinytext	Adresa trvalého bydliska
record_created	datetime	Dátum a čas registrácie do systému

2.2.2 Relácie

Relácie v InnoDB vytvoríme pomocou cudzích kľúčov. Zo štruktúry databázy, popísanej v kapitole 2.2.1, vyplýva zoznam jednotlivých cudzích kľúčov, pričom názvy sĺpcov som volil vzhľadom k tomu, k akej tabuľke sa vzťahujú. V tabuľkách so štruktúrou (viď tab. 2.3 až tab. 2.7) sú to všetky stĺpce označené v popise ako index. Ide však o skutočný index vytvorený i priamo v databázy, pre indexáciu hodnôt a pre ich rýchlejšie hľadanie pri pokladaní SQL dotazov. Výsledná štruktúra databázy je v prílohe A na obr. A.1.

2.2.3 Príznaky záznamov – stav rezervácie

V databázových tabuľkách rezervácií je stĺpec `status`, ktorý nám symbolizuje stav rezervácie. Jedná sa o záznam typu enum, ktorý môže nadobúdať hodnoty:

- Hodnota `new` nám predstavuje záznam, ktorý je v databáze nový, zatiaľ nepotvrdený správcom. Takúto rezerváciu ďalej môže užívateľ upravovať.
- Hodnotu `confirmed` záznam nadobudne, ak správca záznamu potvrdí rezerváciu. Vtedy rezervácia naberá platnosť, a pri rezervácií zariadení sa sprístupní vygenerovanie zapožičiavacieho listu. Užívateľ je o tomto informovaný e-mailom, pričom priamo v jeho texte sa nachádza odkaz na zapožičiavací list.
- Ak správca však zamietne rezerváciu, nastaví sa hodnota `refused`. Vtedy je rezervácia neplatná a užívateľ je opäť informovaný e-mailom. Správca má tiež možnosť udať dôvod zamietnutia, ak takto urobí, tento dôvod sa pripojí k textu e-mailovej správy.

Keďže často dochádza k hľadaniu záznamov práve podľa tohto stĺpca, je v databáze nastavený index na tento stĺpec. Tým máme zaistené rýchlejšie výbery z databázy, ktoré sú často podmienené ešte iným indexom, napr. ID užívateľa.

2.2.4 Príznaky záznamov – odstránený záznam

Aby bolo možné záznamy mazať, pridal som príznak `deleted` každej databázovej tabuľke. Takto máme zaistenú konzistenciu dát[4] a navyše je možné jednoduchou zmenou jednej hodnoty vrátiť zmazaný záznam späť. Keďže pri webových aplikáciách s viacerými administrátormi a správcami záznamov často vznikajú nezhody pri zisťovaní prečo bol nejaký záznam odstránený, kto tak vykonal a pod., doplnil som príznak ďalšími dvomi stĺpcami:

- `deleted_date` – dátum zmazania záznamu
- `deleted_by` – ID užívateľa, ktorý záznam odstránil

Keďže príznak `deleted` sa používa takmer pri každom výbere dát, je naň opäť nastavený index.

2.2.5 Modely

Pri tvorbe Model–View–Presenter webovej aplikácie musíme vytvoriť prístup k dátam v databáze. To sa deje vďaka modelom. Pre tvorbu modelov a ich následné použitie som využil službu[5] `ModelLoader`, ktorú som vytvoril. Táto služba sa stará o automatické načítanie modelov v prípade ich potreby. Samotné modely sú postavené na jednoduchej štruktúre, ktorú som si počas používania frameworku sám vytvoril. Táto štruktúra je založená na dedení vlastností zo základného abstraktného modelu (`BaseModel`), ktorý obsahuje metódy pre jednoduché vyhľadávanie a ukladanie dát. Zároveň sa stará, aby každý model bol správne inicializovaný, tj. mal správne vyplnený názov tabuľky. Pre pokladanie dotazov na databázu využívam open source knižnicu dibi[6], ktorá eliminuje možnosť využitia útoku zvaného SQL injection, tj. injektovanie dotazu do iného dotazu.

Jednotlivé modely sú nazvané podľa názvov databázových tabuliek. Teda je jednoznačné, aké dáta s ktorým modelom dokážeme načítať, pričom niektoré závislosti som – kvôli jednoduchosti použitia a menšej výpočtovej náročnosti na server – implementoval už priamo v metódach samotného modelu, viď tab. A.3.

Setting model

V zozname modelov (príloha A.3) sa nachádza navyše model nazvaný ako `setting`. Tento model sa neviaže na žiadnu databázu. Jedná sa o špeciálny model – objekt, ktorý ako úložisko dát využíva súbor formátu neon. Slúži pre prístup k nastaveniam, ktoré môžu byť administrátormi menené. Keďže tieto dáta sa málokedy menia, zvolil

[4] I keď toto už zaisťuje samotná databáta InnoDB.

[5] Príklad použitia služieb je v teoretickej časti v kapitole 1.2.4.

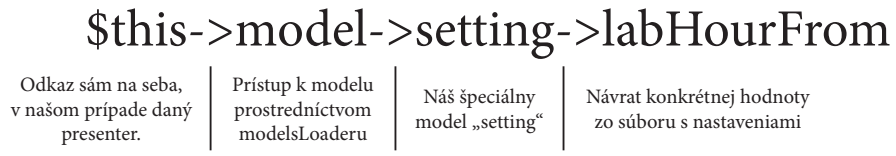
[6] Knižnicu dibi možno stiahnuť z http://www.dibiphp.com/.

som možnosť využiť súborový systém pre ukladanie dát, a teda konkrétne rozšírenie jazyka YAML. Príklad výstupného súboru neon je na obr. 2.1.

```
labHourFrom: "6"
labHourTo: "22"
labMaxWeeks: "12"
email_from: noreply@feec.vutbr.cz
email_regto: admin@email.com
use_smtp: false
smtp: []
```

Obr. 2.1: Príklad súboru neon s nastaveniami.

Objekt sa vytvorí, len ak je požadovaný. Pri vytvorení (v konštruktore) si načíta údaje zo súboru `settings.neon`, pričom využíva bezpečný protokol `safe://`. Toto zabraňuje tomu, aby bol súbor zmenený počas toho ako je otvorený, respektíve načítaný pokiaľ ešte nebol ukončený zápis. K jednotlivým parametrom nastavení nasledovne pristupujeme, ako keby sme pracovali s jednoduchým objektom. Teda ak chceme v niektorom presentery pristúpiť napríklad k nastaveniu hodiny otvorenia laboratorií, použijeme k tomu kód na obr.2.2.



Obr. 2.2: Príklad prístupu k premennej v setting modely.

2.3 Návrh užívateľského rozhrania

2.3.1 Rozdelenie prostredia, presentery

Samotné užívateľské prostredie možno navrhnúť dvoma rôznymi spôsobmi:

- Rozdelenie na časť užívateľskú a na časť administrátorskú, ako to býva pri väčšine klasických webových aplikáciách.[7] Takéto rozdelenie sa však už počas testovania prejavilo ako nepraktické a zbytočne zložité pre koncových užívateľov, hlavne pre správcov a administrátorov, ktorý majú širšie možnosti.
- Jedno rozhranie, pričom sa každému zobrazujú len akcie, ktoré mu jeho užívateľská rola dovoľuje.

Následne nám už zo zadania plynie rozdelenie do troch rôznych rozhraní:

- laboratóriá,
- meracie zariadenia,
- užívatelia.

Každé rozhranie nám predstavuje použitie jedného presenteru, pričom musíme pridať presenter pre akcie: prihlásiť, odhlásiť a registrovať.[8] Výsledný zoznam presenterov je teda:

BasePresenter rieši oprávnenia k jednotlivým presenterom a ich akciám. Umožňuje prístup k modelom v ostatných presenteroch. Tu som tiež vytvoril metódu, ktorá sa stará o automatické načítavanie formulárových a tabuľkových komponentov (Tabella). V metóde `beforeRender` som pridal dodatočné helpery, popísané v kapitole 2.3.5. Okrem iného, v metóde `startup` je vyriešené ukladanie absolútnej adresy, ktorú požaduje neprihlásený užívateľ, aby mohol byť na túto po prihlásení presmerovaný. Toto sa deje napríklad, ak užívateľ použije odkaz z e-mailovej správy (napr. na konrétny zapožičiavací list).

DevicePresenter slúži pre zobrazenie zoznamu meracích zariadení, ich rezerváciu, zobrazenie zoznamu rezervácií daného zariadenia a pre vygenerovanie zapožičiavacieho listu (viď príloha C obr. C.2). V tomto presentery sa tiež spracúvajú informácie odosielané formulárom pre pridanie nových zariadení do databázy, odosielané formulárom pre úpravu zariadení, a tiež z formulára pre pridanie novej rezervácie, pričom validácia je riešená priamo v modeli `Device`.

ErrorPresenter spracováva chybové výnimky, vytvorené už v základnej štruktúre frameworku. Rieši chyby 403, 404, 405, 410, ostatné chyby typu 4xx a 500.

[7] Napríklad pri tvorbe blogu, portfólia apod.

[8] Toto zabezpečuje presenter Sign.

HomePresenter vykresľuje základnú domovskú obrazovku zobrazenú po prihlásení užívateľa. Tu môže užívateľ vidieť svoje rezervácie prístrojov v prehľadnej tabuľke, ktorá obsahuje aj linky na jednotlivé zapožičiavacie listy, a svoje rezervácie miestností. Jednotlivé rezervácie sú farebne zvýraznené podľa stavu ich potvrdenia. Ak sa jedná o nepotvrdenú rezerváciu, riadok je zašednutý, ak sa jedná o zamietnutú rezerváciu, zobrazuje sa červené varovanie. Názvy miestností sú vo forme odkazu a smerujú na stránku s kalendárom danej miestnosti v daný deň rezervácie. Tento presenter obsahuje tiež funkcie pre zmenu registračných informácií a zmenu hesla.

RoomPresenter slúži pre zobrazenie zoznamu laboratórií podľa zvolenej adresy budovy, ďalšiu možnosť ich filtrácie a možnosť pridávania nových laboratórií. Taktiež rieši odosielanie dát pre našepkávač pri pridávaní nových miestností, ajaxové odpovede pre zobrazenie aktuálneho obsadenia danej miestnosti podľa zvoleného dátumu. Presenter spracúva dáta odosielané formulármi pre rezerváciu a pre pridanie nového laboratória do databázy. Správcom a administrátorom umožňuje zobraziť prehľadný kalendár obsadenia danej miestnosti s možnosťou exportovania do PDF súboru.

SettingPresenter umožňuje administrátorom nastavovať jednotlivé vnútorné nastavenia rezervačného systému a tak ovplyvňovať jeho chovanie. Všetky hodnoty sa nastavujú prostredníctvom formulárového komponentu `SettingForm`, ktorý je bližšie popísaný v kap. 2.3.3.

SignPresenter sa stará o správne prihlásenie a odhlásenie užívateľa, o vykreslenie prihlasovacej obrazovky a o spracovanie dát z formulára pre registráciu užívateľov.

UserPresenter je určený len pre administrátorov. Má za úlohu zobraziť tabuľku so zoznamom všetkých registrovaných užívateľov s možnosťou úpravy ich kompletných registračných údajov a možnosťou zmeny užívateľskej role. Užívateľov je možné vyhľadávať a filtrovať podľa všetkých parametrov zobrazených v hlavičke komponentu `UserGrid`, opísaného v kapitole 2.3.4. Tento presenter tiež zaisťuje možnosť vygenerovať užívateľovi nové heslo. Toto je ihneď zašifrované a uložené do databázy, daný užívateľ je zároveň o tomto informovaný prostredníctvom e-mailovej správy. Presenter takisto zaisťuje možnosť zmazania užívateľa z databázy. Toto sa však deje len prostredníctvom nastavenia príznaku `deleted`, aby boli údaje konzistentné, a bolo tak možné dohľadať kompletnú históriu rezervácií, či už sa jedná o prístroje alebo miestnosti.

2.3.2 Užívateľské role

Pre jednoduché nastavenie oprávnení som vytvoril súbor `permissions.neon`. Obsah súboru sa nachádza v prílohe B.1. K tomuto súboru pristupuje základný abstraktný presenter (všetky presentery dedia jeho vlastnosti), ktorý v metóde `startup` (viď obr. 1.2) kontroluje, či má užívateľ oprávnenia k danému presenteru a akcii, a či je užívateľ vlastne prihlásený – ak nie, je presmerovaný na prihlasovaciu obrazovku.

Užívatelia sú rozdelený do štyroch rôznych skupín. Pri registrácií získavajú oprávnenia užívateľskej role GUEST, kedy majú možnosť iba zobrazovať zoznamy laboratórií a meracích zariadení. Ak administrátor takémuto užívateľovi udelí oprávnenia užívateľskej role USER, sprístupni mu možnosť rezervácie. Ak má užívateľ pridelenú rolu MANAGER, teda správca, má možnosť pridávať nové laboratóriá a meracie zariadenia, a spravovať ich. Spravovať môže iba svoje pridané položky. K ostatným má rovnaký prístup ako užívateľská rola USER. Administrátor, teda užívateľská rola ADMIN, má prístup ku všetkým spomínaným akciám a navyše aj k úpravám užívateľov.

2.3.3 Formulárové komponenty

Formuláre sú vykreslované pomocou tried, ktoré som umiestnil do zložky `forms`. Takto možno sprehľadniť kód presenterov, inak by v presenteroch musela byť pre každý formulár metóda navyše, ktorá komponent vytvorí a predá šablóne. O toto sa stará jednoduchá metóda v základnom abstraktnom presentery (viď obr. 2.3).

```
/**
 * Rozsirenie nacitavania komponentov pre automaticke vytvaranie
 * formularov (namespace Forms) a tabuliek (namespace Grids)
 * @param string $name nazov komponentu
 * @return Nette\ComponentModel\IComponent
 */
protected function createComponent($name) {
    if (Strings::endsWith($name, 'Form')) {
        $className = 'Forms\\' . Strings::firstUpper($name);
        $form = new $className($this, $name);
        $form->onSuccess[] = callback($this, 'on' . Strings::firstUpper($name) . 'Submitted');
        return $form;
    }
    elseif (Strings::endsWith($name, 'Grid')) {
        $className = 'Grids\\' . Strings::firstUpper($name);
        return new $className($this, $this->model);
    }
    return parent::createComponent($name);
}
```

Obr. 2.3: Zdrojový kód funkcie pre automatické načítanie komponentov.

Táto metóda nám teda rieši všetky formulárové komponenty a komponenty typu Tabella (nazval som ich ako Grid, odvodené od známeho DataGridu). Formulárovým komponentom automaticky pridáva callback, teda nejaký interný odkaz, ktorý sa má zavolať pri odoslaní formulára. Toto som nastavil na metódu presenteru, ktorú som nazval `on<NázovFormulára>Submitted`. Výsledný zoznam formulárových komponentov je teda:

ChangePasswordForm vytvára formulár, ktorý je prístupný každému registrovanému užívateľovi. Povoľuje mu meniť svoje heslo, pričom kvôli bezpečnosti je vyžadované aj súčasné heslo užívateľa.

DeviceAddForm vytvára formulár pre vytvorenie nového meracieho zariadenia v databázy, pričom pri každej položke sú priradené validačné pravidlá. Tieto sú potom kontrolované pomocou JavaScriptu a následne i na servery v PHP skripte. V prípade zlého vyplnenia položky sa zobrazí chybová hláška, formulár ostane zobrazený a užívateľ je vyzvaný k náprave chyby. Formulárový komponent vyžaduje pre správne uloženie dát základné informácie o meracom prístroji a jeho kompletné evidenčné údaje.

DeviceEditForm vytvára formulár podobný predošlému, avšak obsahuje možnosť zmeniť správcu daného záznamu. Táto možnosť je nevyhnutná pre kompletnú správu záznamu. V prípade že správca daného záznamu zmení túto hodnotu, udelí post správcu záznamu niekomu inému a zároveň týmto stratí možnosť spravovať daný záznam. Táto akcia je nevratná, je nutné kontaktovať administrátora alebo aktuálneho správcu záznamu.

DeviceReservationForm má na starosti zobrazenie formulára pre rezerváciu meracieho zariadenia. Formulár využíva formátovaný vstup údajov, pričom o tento sa stará jQuery plugin nazývaný ako „datepicker“. Užívateľovi sa tak po kliknutí na textové pole dátumu zobrazí miniatúrny kalendár, v ktorom je možné listovať v mesiacoch, a po kliknutí na určitý deň sa vloží daný dátum do textového poľa. Validácia formátu je riešená tak na troch miestach: plugin datepicker, JavaScriptová validácia formuláru a samotná validácia v jazyku PHP – teda priamo v tomto objekte. Ďalšiu validáciu dátumov rieši model, ktorý v prípade nesprávneho vyplnenia vracia výnimku `ModelException` – tú spracováva presenter. Takto vytvorená rezervácia je však uložená ako nová rezervácia (stĺpec `status` je nastavený na `new`). Po úspešnom potvrdení rezervácie je užívateľ vyzvaný k vytlačeniu zapožičiavacieho listu prostredníctvom e-mailovej správy.

LoginForm slúži pre vytvorenie formulára pre prihlásenie. Tento sa zobrazuje na prihlasovacej stránke. Okrem neho má neprihlásený užívateľ prístup ešte k registračnému formuláru.

RegisterForm vytvára formulár so základnými údajmi o užívateľovi, ktoré musí nový užívateľ vyplniť, aby sa mohol zaregistrovať do databázy rezervačného systému. Tu prebieha validácia emailu, zašifrovanie hesla, validácia osobných a kontaktných údajov.

RegisterEditForm je podobný ako predošlí objekt, avšak vygenerovaný formulár neobsahuje položky pre nastavenie hesla. Formulár zabezpečuje iba editáciu základných registračných údajov.

ReservationRefuseForm je objekt, ktorý sa používa na stránke pre zamietnutie rezervácie, či už sa jedná o rezerváciu meracieho zariadenia alebo rezerváciu miestnosti. Formulár nám dáva možnosť informovať užívateľa o dôvode zamietnutia jeho žiadosti o rezerváciu.

RoomAddForm formulár pre pridanie nového laboratória do databázy rezervačného systému. Tu správca vyplní údaje o adrese budovy a informácie laboratóriu. Správcovo ID je uložené do databázy k tejto položke, aby mal k nej prístup a práva správcu. Údaje z úspešne odoslaného formulára sa ukladajú do dvoch tabuliek, teda do adresy budovy (tabuľka `building`) a do laboratórií (tabuľka `room`). Následne sa tieto záznamy prepoja pomocou cudzieho kľúča.

RoomEditForm vytvára formulár s rovnakými položkami ako predošlí objekt, avšak pridáva možnosť zmeniť správcu zariadenia.

RoomReservationForm podobne ako predošlý rezervačný formulár, slúži na rezerváciu laboratória. Užívateľ si vyberie pomocou spomínaného minikaledára (plugin „datepicker“) dátum, a následne časy. Pri zmene dátumu je v šablóne vyriešené odoslanie asynchrónneho ajaxového dotazu[9], pomocou ktorého zistíme a zobrazíme tabuľku obsadenia miestnosti v daný deň. Užívateľ má tak grafické znázornenie obsadenia v daný deň stále pred sebou a môže si vybrať čas ktorý mu vyhovuje. Formulár obsahuje možnosť opakovanej rezervácie – vtedy sa vytvorí viac záznamov rezervácie a užívateľ si tak môže rezervovať danú miestnosť na niekoľko týždňov v rovnakom čase.[10]

[9] Túto funkcionalitu využívam z knižnice jQuery.

[10] Maximálny možný počet týždňov možno nastaviť v nastaveniach systému.

RoomReservationEditForm vytvára zjednodušený formulár, ktorý slúži pre úpravu rezerváciu užívateľom.

SettingForm vytvára formulár určený na nastavenie systému. Odoslané údaje prostredníctvom tohto formulára sú zapísané pomocou špeciálneho modelu `setting`. Formulár umožňuje nastaviť:

- časy otvorenia/zatvorenia laboratórií,
- maximálny počet týždňov pre opakované rezervácie,
- adresu odosielateľa systémových e-mailov[11],
- administrátora, ktorý bude informovaný o nových registráciách užívateľov,
- prípadné využitie SMTP serveru pre odosielanie e-mailových správ.

UserEditForm formulár slúži na úpravu všetkých užívateľových registračných informácií. Formulár môže využívať len administrátor. Tu môže skontrolovať všetky údaje užívateľa a prípadne mu zmeniť užívateľskú rolu.

2.3.4 Tabuľkové komponenty Tabella

Komponenty Tabella slúžia na vykreslenie tabuľky záznamov s možnosťami hľadania a filtrovania. Komponenty som oddelil do samostatného priečinku nazvaného `grids`. O načítanie komponentov sa stará spomínaná metóda presenteru `createComponent` popísaná v kapitole 2.3.3. V triede daného komponentu stačí definovať stĺpce, ktoré má tabuľka obsahovať, a ako sa majú údaje z databázy filtrovať podľa daného stĺpca. Objektu Tabella je ďalej predávaný zdroj dát (`dataSource`) z modelu. Pomocou tohto zdroju sa ďalej generujú dotazy na databázu a zobrazujú výsledky hľadania a filtrácie. Pre využitie dynamickej šírky jednotlivých stĺpcov som však musel originálny komponent Tabella trochu upraviť. Od tejto triedy dedia potom ostatné komponetny v priečinku `grids` všetky vlastnosti. Zoznam komponentov z priečinku:

DeviceGrid definuje tabuľku so zoznamom meracích zariadení, s možnosťami filtrácie podľa evidenčného čísla, názvu, miestnosti a zodpovednej osoby.

RoomGrid definuje tabuľku so zoznamom laboratórií, pričom vyhľadáva vždy iba vo zvolenej budove. Následne je možné filtrovať výsledky podľa čísla dverí, názvu laboratória a zodpovednej osoby.

UserGrid tabuľka so zoznamom užívateľov, možnosti filtrácie podľa loginu, priezviska, e-mailovej adresy a užívateľskej role.

[11] To sú všetky akcie, ktoré sa neviažu na užívateľa.

Pre zabezpečenie potvrdzovania jednotlivých rezervácií nám vznikla požiadavka na ďalšie objekty, pričom objekty majú nastavený základný filter – zobrazujú sa len rezervácie, ktoré doposiaľ nie sú potvrdené. Jedná sa o nasledujúce objekty:

DeviceReservationGrid definuje komponent, pomocou ktorého môžeme zobraziť kompletnú históriu rezervácií, s prehľadom dátumov rezervácií, užívateľov ktorý rezerváciu previedli, dôvodom a komentárom k rezervácií. Ak sa jedná o novú (nepotvrdenú) rezerváciu, zobrazí sa možnosť pre potvrdenie alebo zamietnutie rezervácie, pričom pri zamietnutí rezervácie je možnosť udať dôvod zamietnutia. Tento dôvod sa následne pripojí k e-mailu, a tak je užívateľ presne informovaný, čo sa stalo s jeho rezerváciou.

RoomReservationGrid funguje rovnako ako predošlí komponent, avšak pri jednotlivých rezerváciách má navyše možnosť „zobraziť“. Keďže sa komponent nachádza na rovnakej obrazovke, ako kalendár prehľadu obsadenia danej miestnosti, po kliknutí na možnosť „zobraziť“ sa dynamicky načíta[12] kalendár, pričom sa zobrazí len deň s danou rezerváciou. Takto môže správca danej miestnosti získať rýchly prehľad o rezerváciách v daný deň, a rozhodnúť tak, či danú rezerváciu potvrdí alebo zamietne.

DeviceReservationConfirmedGrid predstavuje obmedzený komponent, slúžiaci pre užívateľov. Zobrazuje len potvrdené rezervácie, pričom neumožňuje spúšťať žiadne akcie a nezobrazuje stĺpec stavu záznamu.

2.3.5 Helpery

Pred vykreslením latte šablóny je nutné zaregistrovať dodatočné helpery. Počas návrhu som sa stretol s troma prípadmi, kedy som potreboval vytvoriť nový helper. Jedná sa o tieto prípady:

- `date` – vypisovanie formátovaného dátumu,
- `time` – vypisovanie formátovaného času,
- `reservationRow` – vykreslenie jedného riadku rezervácie laboratória.

Formátovanie dátumu a času

Pre formátovanie časových údajov využívam PHP funkciu `date`, pomocou ktorej je možné jednoducho formátovať vstupný údaj vo formáte „timestamp“, teda unixový čas – počet sekúnd od 1. 1. 1970.

[12] Riešené pomocou technológie AJAX. Toto zabezpečuje knižnica jQuery.

Helper `date` formátuje dátum pomocou formátu `d.m.Y`, kde `d` je deň v mesiaci s úvodnými nulami (01 až 31), `m` je numerická reprezentácia mesiaca s úvodnými nulami (01 až 12), a `Y` je rok (v plnom tvare). Zdrojový kód tohto helperu je na obr. 2.4.

```
$this->template->registerHelper('date',
function ($db_date) {
    return date('d.m.Y', strtotime($db_date));
});
```

Obr. 2.4: Zdrojový kód helperu date.

Helper `time` pracuje na rovnakom princípe, akurát využíva formát `H:i`, kde `H` je 24-hodinový formát času, konkrétne hodiny s úvodnými nulami (00 až 23), a `i` sú minúty s úvodnými nulami (00 až 59). Zdrojový kód je na obr. 2.5. Vytvorením týchto dvoch formátovacích helperov som pokryl všetky požiadavky na formátovanie dátumu a času vo všetkých šablónach rezervačného systému.

```
$this->template->registerHelper('time',
function ($db_date) {
    return date('H:i', strtotime($db_date));
});
```

Obr. 2.5: Zdrojový kód helperu time.

Kalendár rezervácií laboratórií

Grafické rozhranie kalendáru je vytvorené pomocou HTML tabuľky, pričom jednotlivé riadky tejto tabuľky sú vykreslované helperom `reservationRow`. Tento helper má následovne i druhý vstupný parameter (parameter `$date`), ktorý má za úlohu predať helperu dátum daného dňa, ktorý chceme vykresliť. Helper teda zo všetkých rezervácií (parameter `$reservations`) použije len tie, ktoré súhlasia s daným dátumom.

Helper `reservationRow` rieši aj vykreslenie buniek podľa pracovných a nepracovných dní. Zistí, aký je možný rozsah hodín v daný deň, kedy je možné laboratórium rezervovať, a následne vykresľuje po hodinách jednotlivé bunky riadku tabuľky. Tieto bunky môžu byť prázdne (ak nie je registrovaná žiadna rezervácia v danú hodinu), vyplnené konkrétnym dôvodom rezervácie a menom užívateľa, alebo

sa vytvorí šedá bunka (premenná `$gray_cell`), kedy sa zobrazuje bunka s nápisom „Pravděpodobně obsazeno“. Takto je užívateľ informovaný, že už niekto v danú dobu požiadal o rezerváciu, ale správca mu ju doposiaľ nepotvrdil. Takto sa užívateľ dokáže vyhnúť prípadnému odmietnutiu rezervácie z dôvodu, že niekto iný požiadal o rezerváciu skôr. Zdrojový kód helperu je v prílohe na obr. B.1.

Tento helper sa tiež používa pri zobrazení aktuálneho obsadenia miestnosti pri formulári rezervácií laboratórií. Pri zmene dátumu vo formulári sa dynamicky načíta iba element kalendára – zmení sa na vybraný deň. Zdrojový kód použitia tohto helperu v šablóne sa nachádza na obr. 2.6.

```
<table class="room-graph room-graph-simple styled">
    <tr>
        <th n:for="$h = $h_from; $h < $h_to; $h++">{$h}:00</th>
    </tr>
    <tr>
        {!$reservations|reservationRow:$sel_date}
    </tr>
</table>
```

Obr. 2.6: Použitie helperu reservationRow v šablóne.

2.3.6 PDF výstup

V predošlých kapitolách som zmienil výstup do PDF súboru, napríklad pri exportovaní kalendára obsadenia laboratória. O to sa stará trieda `PdfResponse`, ktorá využíva open source knižnicu `mPDF`. Trieda `PdfResponse` bola však určená pre staršiu verziu frameworku Nette, takže som bol nútený prepísať túto triedu pre aktuálnu verziu a odladiť chyby, s ktorými som sa stretol. Príklady systémom vygenerovaných PDF dokumentov sú v prílohe C.

2.3.7 Informačné e-maily

Pre vytvorenie e-mailovej správy využívam latte šablóny, ktoré sú uložené v zložke `templates/_emails`. V šablóne je možné využívať všetky makra, helpery a pod., teda vytvoriť podmienku priamo v šablóne nie je žiaden problém. Na obrázku 2.7 sa nachádza príklad šablóny.[13] Zoznam všetkých šablón sa nachádza v tab. 2.8.

[13]Konkrétne sa jedná o šablónu používanú pre informovanie užívateľa o zamietnutí jeho rezervácie miestnosti, pričom dôvod zamietnutia sa zobrazí iba ak ho správca vyplní.

```
Vaše rezervace místnosti {$room->door_number} byla zamítnuta.

{if !empty($purpose)}
Důvod zamítnutí:
{$purpose}

{/if}
V případě jakýchkoliv dotazů, kontaktujte odesílatele emailu.
```

Obr. 2.7: Príklad šablóny e-mailovej správy.

Vygenerovaný text e-mailovej správy sa predáva objektu `Nette\Mail\Message`, ktorému sa ďalej nastavia adresa odosielateľa, adresa prijímateľa a predmet správy. Ak je v nastaveniach povolený SMTP server, predajú sa objektu tieto nastavenia a nakoniec sa správa odošle.

Tab. 2.8: Zoznam šablón e-mailových správ.

Názov šablóny	**Príjemca**	**Odoslanie pri akcií**
password	Užívateľ	Vygenerovanie nového hesla do systému
newRegistration	Administrátor	Do systému pribudol nový užívateľ
newRoomReservation	Správca miestnosti	Nová žiadosť o rezerváciu miestnosti
reservationConfirmedRoom	Užívateľ	Potvrdenie žiadosti o rezerváciu miestnosti
reservationRefusedRoom	Užívateľ	Žiadosť o rezerváciu miestnosti bola zamietnutá
newDeviceReservation	Správca zariadenia	Nová žiadosť o rezerváciu prístroja
reservationConfirmedDevice	Užívateľ	Potvrdenie žiadosti o rezerváciu prístroja
reservationRefusedDevice	Užívateľ	Žiadosť o rezerváciu prístroja bola zamietnutá

3 ZÁVER

Výsledkom mojej práce je rezervačný systém s jednoduchými možnosťami administrácie. Položky v rezervačnom systéme je možné zobrazovať v zozname s možnosťami hľadania a filtrácie, pričom administrátori a správcovia majú možnosť kompletnej správy záznamov. Systém zatiaľ beží na testovacom webhostingu, pričom je vytvorený návod pre inštaláciu na iný webový server. Systém obsahuje plnohodnotnú registráciu nových užívateľov, pričom je nutné užívateľovi prideliť vyššiu užívateľskú rolu pre prístup k akciám zapisovania údajov do rezervačného systému. Všetci užívatelia majú prístup k detailnému kalendáru obsadenia laboratórií, pričom zobrazenie rezervácií prístrojov je navrhnutý ako filtrovateľný a prehľadávateľný zoznam. O všetkých zmenách rezervácií sú užívatelia informovaný prostredníctvom informačných e-mailov.

Systém využíva jazyk PHP a databázový systém MySQL. Bol navrhnutý tak, aby jeho ďalšie rozširovanie bolo čo najjednoduchšie.

LITERATÚRA

[1] GRUDL, D. *Dokonalé zabezpečení webových aplikací* [online]. 2010, posledná aktualizácia 15. 2. 2010. Dostupné z URL: <http://nette.org/cs/hlavni-prednosti/bezpecnost>.

[2] ŠIMŮNEK, M. *SQL kompletní kapesní průvodce*. Praha: GRADA Publishing, 1999. 248 s. ISBN 80-7169-692-7.

[3] BERT, A. C. *InnoDB*. Chromo Publishing, 2011. 76 s. ISBN 978-6136581002.

[4] YANG, Y. *MySQL Engines: InnoDB vs. MyISAM — A Comparison of Pros and Cons* [online]. 2009, posledná aktualizácia 2. 9. 2009. Dostupné z URL: <http://www.kavoir.com/2009/09/mysql-engines-innodb-vs-myisam-a-comparison-of-pros-and-cons.html>.

[5] SKŘIVAN, J. *Databáze a jazyk SQL* [online]. 2000, posledná aktualizácia 4. 8. 2000. Dostupné z URL: <http://interval.cz/clanky/databaze-a-jazyk-sql/>.

[6] KULHAN, J. *Normalizace relačních databází* [online]. 2008, posledná aktualizácia 23. 7. 2008. Dostupné z URL: <http://programujte.com/clanek/2008071900-normalizace-relacnich-databazi/>.

[7] STOUPA, V. *Přehled a vývoj PHP frameworků* [online]. 2008, posledná aktualizácia 28. 3. 2008. Dostupné z URL: <http://www.root.cz/clanky/prehled-a-vyvoj-php-frameworku/>.

[8] *Framework* [online]. 2006, posledná aktualizácia 14. 6. 2011. Dostupné z URL: <http://cs.wikipedia.org/wiki/Framework>.

[9] TVRDÍK, J. *MVC aplikace & presentery* [online]. 2011, posledná aktualizácia 16. 11. 2011. Dostupné z URL: <http://doc.nette.org/cs/presenters>.

[10] *Understanding Model–View–Controller* [online]. 2011, citované dňa 21. 11. 2011. Dostupné z URL: <http://book.cakephp.org/2.0/en/cakephp-overview/understanding-model-view-controller.html>.

[11] McARTHUR, K. *Pro PHP: Patterns, Frameworks, Testing and More*. 1 edition. Apress, 2011. 376 s. ISBN 978-1590598191.

[12] PURCHART, V. *Dependency Injection: motivace* [online]. 2011, posledná aktualizácia 13. 6. 2011. Dostupné z URL: <http://zdrojak.root.cz/clanky/dependency-injection-motivace/>.

[13] GRUDL, D. *Dependency Injection* [online]. 2011, posledná aktualizácia 16. 8. 2011. Dostupné z URL: <http://doc.nette.org/cs/dependency-injection>.

[14] TVRDÍK, J. *Šablony* [online]. 2011, posledná aktualizácia 18. 6. 2011. Dostupné z URL: <http://doc.nette.org/cs/templating>.

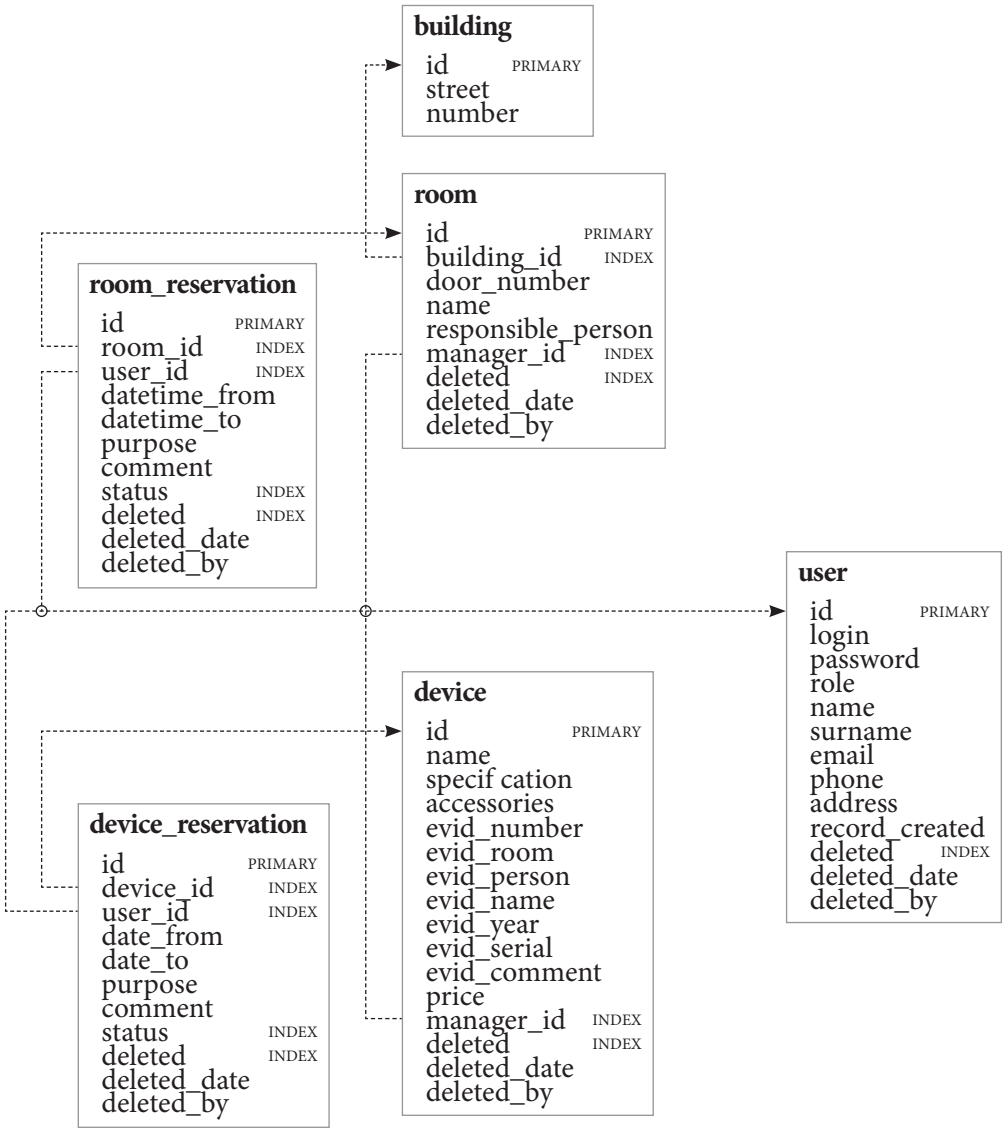
ZOZNAM SYMBOLOV, VELIČÍN A SKRATIEK

1NF	Prvá normálová forma
2NF	Druhá normálová forma
3NF	Tretia normálová forma
BCNF	Boyce-Coddová normálová forma
CSRF	Cross-site request forgery
DB	Databáza
DI	Dependency Injection
HTML	Hypertextový značkový jazyk (HyperText Markup Language)
ID	Identifikačné číslo
JSON	JavaScript Object Notation
PHP	Hypertext Preprocessor
MVC	Model–View–Controller
MVP	Model–View–Presenter
OOP	Objektovo orientované programovanie
SMTP	Simple Mail Transfer Protocol
SQL	Štruktúrovaný dotazovací jazyk (Structured Query Language)
URL	Jednotkový lokátor zdrojov (Uniform Resource Locator)
XML	Rozšíriteľný značkovací jazyk (Extensible Markup Language)
XSL	Rozšíriteľný štýlovací jazyk (Extensible Stylesheet Language)
XSS	Cross-site scripting
YAML	Yet Another Markup Language

ZOZNAM PRÍLOH

A ŠTRUKTÚRA DATABÁZY

A.1 Štruktúra a relácie medzi tabuľkami



Obr. A.1: Štruktúra databázy

A.2 Zoznam cudzích kľúčov

Tab. A.1: Priradenie cudzích kľúčov

Tabuľka	Zdroj	Cieľová tabuľka	Stĺpec
device	manager_id	user	id
device_reservation	device_id	device	id
device_reservation	user_id	user	id
room	building_id	building	id
room	manager_id	user	id
room_reservation	room_id	room	id
room_reservation	user_id	user	id

A.3 Zoznam modelov

Tab. A.2: Metódy základného abstraktného modelu

Model	Metóda	Popis metódy
BaseModel	___construct	Konštruktor, kontrola nastavenia názvu tabuľky.
	save	Uloženie dát.
	insert	Vloženie dát do databázy.
	update	Uloženie zmien v databázy.
	findBy	Nájdenie dát podľa zadaného stĺpca a hodnoty.
	getDataSource	Vracia zdroj dát pre objekt Tabella.

Tab. A.3: Zoznam modelov a ich metódy

Model	Špeciálna metóda	Popis metódy
Building	getAutocomplete	Dáta pre našepkávač adresy budovy.
Device	getManagersDropDown	Dáta pre filtrovanie prístrojov podľa ich správcov.
DeviceReservation	formatDateTime	Formátovanie dátumu pre správne uloženie v DB.
	add	Pridanie rezervácie a test validity dát.
	findAllByDate	Výber rezervácií prístroja v danom období.
	firstDate	Prvý dátum rezervácie zariadenia.
	findAllActiveByUserId	Zoznam aktívnych rezervácií daného užívateľa.
Room	save	Uloženie dát.
	getBuildingId	Zistenie/vytvorenie záznamu adresy laboratória.
	getBuildingsDropDown	Dáta pre filtrovanie laboratórií podľa adresy budovy.
	getManagersDropDown	Dáta pre filtrovanie laboratórií podľa ich správcov.
	getDataSourceByBuilding	Vracia zdroj dát pre objekt Tabella.
RoomReservation	formatDateTime	Formátovanie dátumu pre správne uloženie v DB.
	add	Pridanie rezervácie a test validity dát.
	findAllByDate	Výber rezervácií laboratórií v danom období.
	firstDate	Prvý dátum rezervácie laboratória.
User	getDataSource	Vracia zdroj dát pre objekt Tabella.

A.4 Zdrojový kód základného modelu

```
<?php

namespace Models;

use Nette\ArrayHash,
    dibi;

abstract class BaseModel {

    public function __construct() {
        $class = get_class($this);
        if (empty($class::$table)) {
            throw new ModelException(
             "Public variable 'table' is undefined on model $class!", 500
            );
        }
    }

    public function save(ArrayHash $values) {
        if (!isset($values->id)) {
            return $this->insert($values);
        }
        else {
            return $this->update($values);
        }
    }

    public function insert(ArrayHash $values) {
        $class = get_class($this);
        return dibi::query('INSERT INTO [' . $class::$table . ']', (array) $values);
    }

    public function update(ArrayHash $values) {
        $class = get_class($this);
        return dibi::query('
         UPDATE [' . $class::$table . ']
         SET', (array) $values, '
         WHERE [id] = %i', $values->id
        );
    }

    public function findBy($by, $value, $fields = '*') {
        $class = get_class($this);
        return dibi::fetch("
            SELECT $fields
            FROM [" . $class::$table . "]
            WHERE [$by] = %s", $value
        );
    }

    public function getDataSource() {
        $class = get_class($this);
        return dibi::dataSource($class::$table);
    }

}
```

B UŽÍVATEĽSKÉ ROZHRANIE

B.1 Nastavenie oprávnení

```
GUEST:
    Home: [default, editProfile, changePassword]
    Room: [default, calendar, calendarpdf, calendarOtherUsers]
    Device: [default, list, listpdf, listOtherUsers]
    Sign: [out]

USER < GUEST:
    Room: [reservation, reservationEdit]
    Device: [reservation, reservationEdit, contractpdf]

MANAGER < USER:
    Room: [add, edit, delete, autocomplete, confirmReservation,
           refuseReservation, removeReservation]
    Device: [add, edit, delete, confirmReservation, refuseReservation,
             removeReservation]

ADMIN < MANAGER:
    User: [default, edit, delete, generatePassword]
    Room: [editOtherUsers, deleteOtherUsers, confirmReservationOtherUsers]
    Device: [editOtherUsers, deleteOtherUsers, confirmReservationOtherUsers,
             contractpdfOtherUsers]
    Setting: [default]
```

B.2 Dodatočné helpery

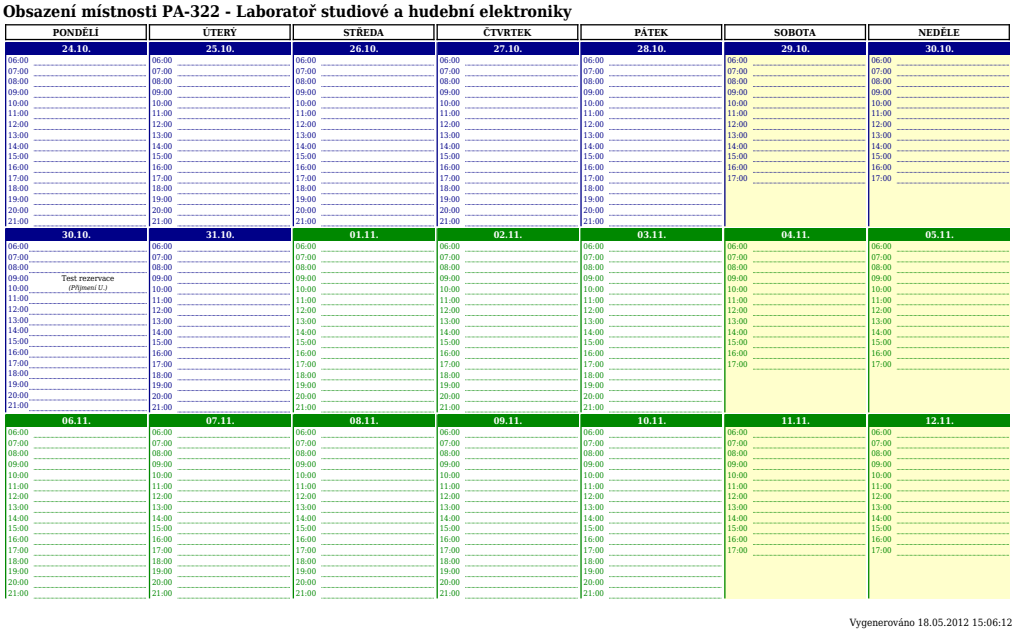
```
$setting = (object)$this->model->setting->allParams;

$this->template->registerHelper('reservationRow',
function ($reservations, $date) use ($setting) {
    $html = '';
    if (is_string($date)) $date = strtotime($date);
    $h_from = date('N',$date) > 5 ? $setting->labHourFromWeekend : $setting->labHourFrom;
    $h_to = date('N',$date) > 5 ? $setting->labHourToWeekend : $setting->labHourTo;
    for ($h = $h_from; $h < $h_to; $h++) {
        $time = ONE_HOUR;
        $info = FALSE;
        foreach ($reservations as $res) {
            $datetime_from = strtotime($res->datetime_from);
            $datetime_to = strtotime($res->datetime_to);
            $hour = (int) date('G', $datetime_from);
            if ($hour == $h && $datetime_from >= $date && $datetime_from <= $date+ONE_DAY) {
                $time = $datetime_to - $datetime_from;
                $info = $res;
                $h += ($time/ONE_HOUR) - 1;
            }
        }
        $html_td = Html::el('td', array(
            'colspan' => $time/ONE_HOUR,
            'class' => $info ? (($info->status=='confirmed')?'red':'gray') : NULL
        ));
        if ($info) {
            if ($info->status == 'confirmed') {
                $purpose = Html::el('div', array('class' => 'purpose'));
                $purpose->setHtml($info->purpose);
                $html_td->add($purpose);

                $user = Html::el('div', array('class' => 'user'));
                $user->setHtml($info->name.' '.$info->surname);
                $html_td->add($user);
            }
            else {
                $gray_cell = Html::el('div', array('class' => 'na'))->add('Pravděpodobně<br />obsazeno');
                $html_td->add($gray_cell);
            }
        }
        else $html_td->setHtml(' '); // aby IE vykreslil prazdne bunky tabulky
        $html .= $html_td;
    }
    return $html;
});
```

Obr. B.1: Zdrojový kód helperu reservationRow.

C PRÍKLADY PDF VÝSTUPOV



Obsazení místnosti PA-322 - Laboratoř studiové a hudební elektroniky

PONDĚLÍ	ÚTERÝ	STŘEDA	ČTVRTEK	PÁTEK	SOBOTA	NEDĚLE
24.10.	25.10.	26.10.	27.10.	28.10.	29.10.	30.10.
30.10.	31.10.	01.11.	02.11.	03.11.	04.11.	05.11.
Test rezervace *(Příjmení U.)*						
06.11.	07.11.	08.11.	09.11.	10.11.	11.11.	12.11.

Vygenerováno 18.05.2012 15:06:12

Obr. C.1: Kalendár obsadenia laboratória

Zápůjčka zařízení

Vystavil: Jiří Schimmel

Purkyňova 118 Brno 612 00	Tel.:	541149167
	E-mail:	schimmel@feec.vutbr.cz

Odběratel:

Jméno Přijmení	Tel.:	
	E-mail:	

Zařízení	**cena s DPH**
Všesměrový zdroj zvuku Příslušenství: stativ, kabel speakon, přepravní rack zesilovače, přepravní box	40 000,00
Cena celkem s DPH:	**40 000,00 kč**

Zapůjčeno za účelem: Testovanie rezervácie zariadenia
Délka zápůjčky: od **6.12.2011** do **16.12.2011**

Způsob vrácení: osobně
na Purkyňova 118, Brno, 612 00
Datum vrácení: 16.12.2011

Potvrzení odběratele:
Převzal dne: 5.12.2011

Souhlasím s podmínkami zápůjčky. **Podpis**

Zboží je majetkem VUT v Brně. V případě znehodnocení či poškození zapůjčeného zařízení se odběratel zavazuje provést neprodleně na vlastní náklady opravu zařízení. V případě zcizení zařízení se odběratel zavazuje uhradit plnou cenu nového shodného zařízení, či zařízení ekvivalentních parametrů.

Obr. C.2: Zapožičiavací list

D DÁTOVÝ DISK

D.1 Popis zložiek a súborov

V koreňovom adresáry priloženého CD disku sa nachádza:

- súbor `navod.txt`, ktorý obsahuje návod na inštaláciu – nájdete tiež v prílohe D.2,
- súbor `tables.sql`, ktorý obsahuje kompletnú databázovú štruktúru i s vytvoreným užívateľom „admin“,
- priečinok `rezervacny_system`, ktorý obsahuje zdrojové súbory samotného rezervačného systému. Tento ďalej obsahuje:
  - `app` priečinok, teda všetky súbory samotnej webovej aplikácie:
    - `forms` obsahuje PHP súbory formulárových komponentov,
    - `grids` obsahuje PHP súbory komponentov Tabella,
    - `models` obsahuje PHP súbory modelov,
    - `presenters` obsahuje PHP súbory presenterov,
    - `templates` obsahuje latte šablóny,
    - `.htaccess` zaisťuje zabezpečenie priečinku app,
    - `bootstrap.php` spustenie dôležitých súčastí frameworku,
    - `config.neon` konfiguráciu frameworku a konfiguráciu prístupu k databáze,
    - `permissions.neon` konfiguráciu prístupových pravidiel pre jednotlivé užívateľské role,
    - `settings.neon` dáta pre špeciálny model „setting“, musí byť povolené zapisovanie do tohto súboru,
  - `libs` obsahuje všetky použité knižnice:
    - `Models`, teda `ModelsLoader` a triedu základného modelu,
    - `PdfResponse` určenú na generovanie PDF výstupov, obsahuje knižnicu mpdf,
    - `Tabella`, základnú triedu všetkých komponentov zo zložky `grids`,
    - `dibi.min.php`, minimalizovaná verzia knižnice dibi vo verzií 2.0 z dňa 3. 2. 2012,
    - `nette.min.php`, minimalizovaná verzia frameworku Nette vo verzií 2.0 z dňa 3. 2. 2012,
    - a súbor `.htaccess` zaisťuje zabezpečenie priečinku libs,
  - `log` obsahuje záznamy chýb a varovaní, zabezpečuje framework,
  - `temp` obsahuje dočasné súbory, musí mať povolený zápis, takisto všetky jeho podzložky,

- `www` je hlavný adresár webu, obsahuje:
  - `adminer` – nástroj na jednoduchú správu databázy,
  - `css` zložku, ktorá obsahuje jednotlivé kaskádové štýly webu,
  - `checker` – nástroj učený pre skontrolovanie požadovanej konfigurácie webového serveru, odporúča sa spustiť pri inštalácií rezervačného systému na server,
  - `images` zložku s obrázkami,
  - `js` zložku so súbormi javasciptu, knižnicou jQuery, a jej rozšíreniami,
  - `.htaccess` súbor, ktorý zabezpečuje presmerovanie všetkých požiadaviek na `index.php`,
  - `favicon.ico` ikonu webu,
  - `index.php` súbor, ktorý obhospodaruje všetky požiadavky, volá súbor `bootstrap.php`, ktorý štartuje samotnú webovú aplikáciu.

D.2 Návod na inštaláciu

1. Nahráme všetky súbory zo zložky `rezervacny_system` na server tak, aby požiadavky na spracovanie smerovali do zložky `www`, v prípade webového serveru Apache je toto možné zaistiť konfiguráciou premennej virtuálneho hosta `DocumentRoot`, v prípade iných webových serverov (napr. Nginx, Cherokee, Lighttpd) je nutné smerovať požiadavky priamo na súbor `www/index.php`.
2. Povolíme zápis do zložky `temp` a jej podzložiek, do zložky `log`, a do súboru `app/settings.neon`.
3. Spustíme priložený nástroj „checker“[1]. Týmto zistíme aké nedostatky má náš server, a čo je nutné prípadne doinštalovať.
4. Vytvoríme štruktúru databázy. Stačí spustiť kód zo súboru `tables.sql` ako SQL dotaz na databázu. K tomu nám pomôže „phpMyAdmin“ na servery, prípadne môžeme použiť nástroj „adminer“[2]. Pozor, databáza musí byť vytvorená v kódovaní UTF–8!
5. Upravíme súbor `app/config.neon`, konkrétne nastavenia databázy v sekcií `production`.
6. Prihlásime sa do rezervačného systému pomocou užívateľa „admin“ s heslom „password“. Užívateľa nemusíme upraviť, ale určite mu zmeníme heslo.

[1] Napr. http://www.example.com/checker
[2] Napr. http://www.example.com/adminer